Bigdata architecture for large-scale scientific computing

Benoit Lange, Toan Nguyen

Bigdata architecture for large-scale scientific computing

Benoit Lange
Project OPALE
INRIA Grenoble
38334 Sant-Ismier, France
benoit.lange@inria.fr

Toan Nguyen
Project OPALE
INRIA Grenoble
38334 Sant-Ismier, France
toan.nguyen@inria.fr

***Abstract*—ABDA'14: POSITION PAPER.**

Today, the scientific community uses massively simulations to test their theories and to understand physical phenomena. Simulation is however limited by two important factors: the number of elements used and the number of time-steps which are computed and stored. Both limits are constrained by hardware capabilities (computation nodes and/or storage).

From this observation arises the VELaSSCo project [1]. The goal is to design, implement and deploy a platform to store data for DEM (Discrete Element Method) and FEM (Finite Element Method) simulations. These simulations can produce huge amounts of data regarding to the number of elements (particles in DEM) which are computed, and also regarding to the number of time-steps processed. The VELaSSCo platform solves this problem by providing a framework fulfilling the application needs and running on any available hardware.

This platform is composed of different software modules: a Hadoop distribution and some specific plug-ins. The plug-ins which are designed deal with the data produced by the simulations. The output of the platform is designed to fit with requirements of available visualization software.

I. Introduction

The data production rate has followed a path similar to computation hardware (based on Moores law). The amount of information has an exponential growth while hardware storage capabilities does not follow a similar path. Moreover, the data produced has also an impact on which architecture is needed. This amount of data is extracted by several sources: sensors, simulations, users, etc. For example, the LSST produces 30 terabytes of astrophysics data every night [1]. Simulations can also create large amount of data as in [2], where the authors present a parallel implementation of the Denhen algorithm [3], an astrophysical N-body simulator. This implementation produces 500 Megabytes of data in 1.19 seconds (for a plummer distribution with 10M particles, and only one time-step). HPC facilities, which are used by scientists to perform simulations, are not currently designed to store such important amounts of data: these systems are only suitable to provide efficient computation capabilities.

This paper presents the VELaSSCo project: it provides a BigData architecture to store the data produced by various simulation engines. This data must is visualized by specific tools. For this purpose, two visualization software are targeted: GID [2] from CIMNE [3] and I-FX [4] from Fraunhofer IGD[5].

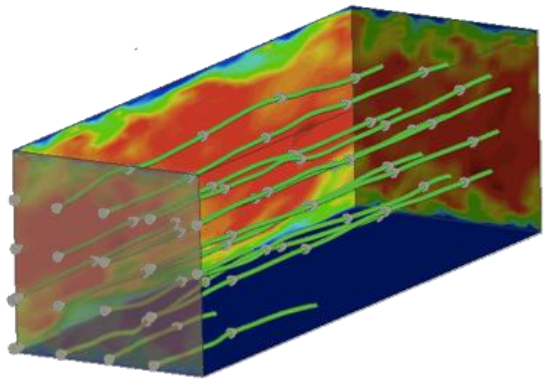

Fig. 1. Visualization of FEM simulation (Air flow), produced by GID (CIMNE).

The project is also focused on specific data produced by two different simulation engines: FEM and DEM data. An example of visualization of FEM simulation data is presented in Figure 1. This simulation deals with the decomposition of space using a mesh structure, and it is used to understand the dynamic of specific objects. For the DEM, a particle example is presented in Figure 2 (The figure has be produced by the University of Edinburg [6]). Both of these solutions produce important amounts of information: for 10 millions particles and 1 billion of time-steps, DEM uses 1 Petabytes of data, or 1 billion elements with 25000 time-steps, whereas FEM produces 50 TB of data. Currently, all the data produced by these simulations data are simply not stored, and several time-steps are deleted from storage device.

This paper focuses on the Big Data architecture designed for the VELaSSCo project. The platform is designed to be scalable regarding to which IT capabilities are available (HPC, Clouds, etc.). It also interfaces visualization tools. It must also interface some commercial tools specifically designed to deal with engineering data.

[1] http://www.velassco.eu

[2] http://www.gidhome.com
[3] http://www.cimne.com
[4] http://www.i-fx.net
[5] https://www.igd.fraunhofer.de
[6] http://www.ed.ac.uk

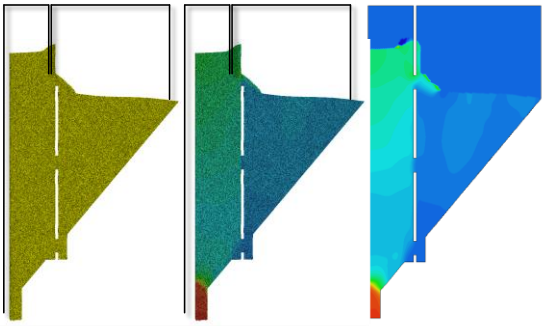

Fig. 2. Visualization of DEM data (UEDIN), silo discharge.

Section II is an overview of related work on Big Data. Section III addresses the architecture of the VELaSSCo platform. Section IV is a conclusion.

II. Related Work

This section is mainly focused on Big Data for engineering applications. Problematics linked with this field are not widely developed in the literature. Most of Big Data related problems concentrate on Web crawling and analytics. Further, simple visualization queries for engineering simulations are similar to Web crawling. Therefore, we assume that using solutions provided for Web search can enhance engineering applications and visualization queries.

MapReduce computation has been massively studied and developed recently. Traditional Big Data approaches are mainly based on MapReduce computations to extract information. Strict implementations have been proposed for this computation model. But evolutions have also been presented and follow two different paths: Hadoop compliant and none Hadoop compliant software. Hadoop [7] is an open-source project which implements all the needs with respect to distribute processing systems for large-scale data. This project was mainly inspired by Google papers [4] and [5].

At the same time, non-Hadoop compliant solutions have been developed, which have been designed by database providers, e.g., to propose a BigData platform based on existing products. In other cases, these solutions are developed to deal with other requirements than Hadoop does. An example is to store big data on HPC facilites without dedicate storage, and run MapReduce jobs on the HPC nodes. These strategies have been designed to provide solutions for running big data applications on traditional data-centers.

Also, the MapReduce programming model has been ported to HPC facilities, while Hadoop is mainly developed to run on a dedicated storage nodes. In [6] and [7], authors present two implementations of MapReduce dedicated to HPC facilities. Their strategies allow to apply MapReduce jobs on POSIX compliant file systems, and an abstract layer is not necessary (like HDFS). A deeper study of these methods is not possible, because the code source is not directly available and the extensibility of these solutions is not discussed.

[7] http://hadoop.apache.org

Global frameworks like Hadoop have also been proposed by the scientific community. One of them is Dryad, [8]. This solution is designed to extend the standard MapReduce model by adding intermediate layers between the Map phase and the Reduce phase. Now, this implementation has been ported to the Hadoop ecosystem, and Dryad is a full extension of Hadoop using YARN. This software is available on the GitHub repository at Microsoft[8].

Regarding our needs, our interest is focused on the Hadoop ecosystem and more precisely on two extensions. The first one deals with the usage of Hadoop over HPC, and the second one deals with an extension of the Hadoop storage with an existing database system.

The paper [9] presents how Hadoop is used over a traditional HPC system. This solution is decomposed as follows: Hadoop services are started, then the necessary files are transferred to HDFS, then the computation is run. After the computation, Hadoop services are stopped and the HDFS partition is destroyed. This solution highlights some bottlenecks: data transfers between HDFS and the HPC file system. Due to the HPC structure, authors do not use local storage of the HPC: indeed this storage can only be used as a temporary repository.

The second paper [10] presents a Hadoop extension which uses a RDBMS (Relational Data Base Management System) to store the data. This storage system is used instead of the traditional HDFS solution. The goal is to improve the query speed over Hadoop using the SQL engine of the RDBMS. Their example stores data into a Postgresql database, but any database system can be used instead.

III. Architecture

This section presents the architecture used for the VELaSSCo platform. It is designed to fit with specific requirements of engineering data simulations. These requirements are:

- the platform has to be compatible with various computing infrastructure: HPC, clouds, grids, etc.,
- the data produced by the simulation engines can be computed by several nodes,
- the visualization queries can be simple or complex,
- the visualization queries will be performed in batch or in real-time,
- the architecture has to be extensible, scalable and supported by a large community of users.

For the first requirement, we are currently extending the solution presented in [9]. This tool provides a solution to deploy a Hadoop ecosystem on any kind of computation infrastructure, and moreover it reduces the bottleneck due to numerous file transfers between the virtual file system (FS) and the HPC FS. Regarding to our partners requirements, it is necessary to provide a solution which can be parameterized to deal with the specifics of their computing facilities. This solution is designed to be suitable for three different cases:

[8] https://github.com/MicrosoftResearchSVC/Dryad

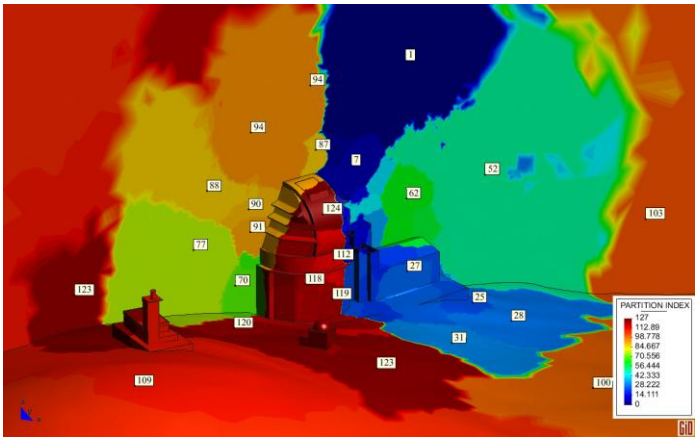

Fig. 3. Different partition of space for a FEM simulation (provided by CIMNE).

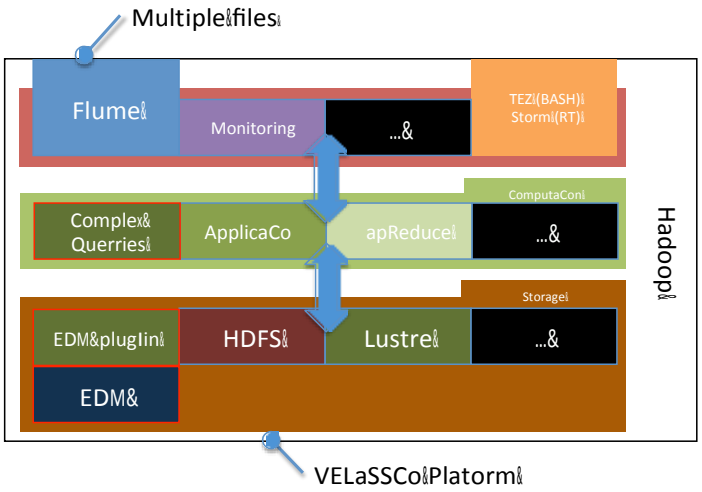

Fig. 4. Expected architecture of the VELaSSCo platform.

1) HPC and dedicated storage nodes,
2) HPC nodes with dedicated local storage,
3) HPC nodes with an existing distributed storage system.

For the first point, a HPC infrastructure coexists with storage facilities. This solution is quite new for users from data simulation. It implies to have two data-centers topologies with dedicated nodes for both sides (HPC and Hadoop). To avoid deployment of such an architecture, it is possible to extend an existing datacenter using external providers like Amazon (with EC2 and S3).

The second approach uses dedicated storage for storing data. All the nodes in the HPC have a specific local storage dedicated to Big Data. A local hard disk is already used to store local data during computations. For these HPC facilites, it is possible to add a specific storage. In this architecture, we dedicate this local storage to all necessary information concerning the BigData architecture. This solution can only be implemented on private computing facilities, with possible hardware modifications.

The last solution uses the distributed FS of the actual HPC to store the data. This approach is the most suitable solution for public computing facilities without extensibility for users. With this approach the data transfers are an important bottleneck.

To fit with all these cases, we extend the myHadoop implementation [9], by providing all the necessary modules to deploy a VELaSSCo platform on all kind of computing facilities. This tool also provides the necessary interfaces to deal with visualization queries, using pre-installed extensions.

The second step of our project is to gather information from computation nodes. The computation of a specific job can imply splitting the data among different nodes: for example FEM simulations decompose the space into elements, which are distributed among the nodes. A representation of decomposition is presented in Figure 3, where each colored area is assigned to a particular node. Thus, for each time-step, it is necessary to gather all the information produced by each node. For this purpose, we use an Apache Flume agent which is in charge of storing information into the VELaSSCo platform.

The third and fourth points concern visualization, and more precisely queries. Visualization has two query layers: the simple one and the complex one.

Simple queries are very similar to traditional information search over Big Data sets. A query has to find a specific subset of information at a specific time-step. This model is well-known and can be efficiently translated into MapReduce jobs. To reduce the complexity of the query model (avoid to define the MapReduce jobs), we use Hive with Tez. But we also have to deal with more complex queries which imply complex computations. For this, specific scripts are developed. Examples of these computations are: extract spline, iso-surface, interpolate information, provide a multi- resolution models, etc. The fourth point concerns the queries rate: queries have to be performed in batch (SQL is well suited for this specific case), but queries can also be triggered dynamically from specific visualization points of view. Displacements of the camera in the 3D space thus produce a queries sent to the platform. For this specific case, we use Storm to stream the data. Different approaches have been proposed in the computer graphic literature, one of them is presented in [11]. This solution presents a continuous multi-resolution method for terrain visualization. Information is sent to the viewer in real-time depending on the camera location. To use efficiently this method, it is necessary to store data using a multi-level approach. In VELaSSCO, we plan to store the data at different resolutions to provide real-time answers to the visualization software. This decomposition of data will be inspired by the method presented by Hoppe in [12], where a base mesh is used to encode all information related to a higher resolution.

The storage architecture of the VELaSSCo platform has tobdeal with this multi-resolution characteristics and hierarchical decomposition. Moreover, the computational model used to extract information has also to be suitable with these assets. This part of the project is the trickier part, and most of our future contributions will be focused on these specific points.

The last point concerns the extensibility and support. We are looking for an extensible framework which supports extensions for specific usage: queries, data locality and the management of specific storage. Hadoop is the best choice for this purpose. The framework already provides a large set of extensions, and scientific communities continuously provide new contributions. Moreover, this solution is well suited to our needs: we provide a plugin to store data into a partner database named EDM (Express Data Manager). The plug-in is inspired by the solution presented in [9]. The EDM database is an object-oriented database designed to store AP209 standard compliant files. It is a database dedicated to engineering applications.

To summarize the whole VELaSSCo platform is depicted in the Figure 4. It enhances the myHadoop software, with preinstalled plugins. It can be deployed on various IT architectures. This solution has been designed to store data from multiple sources using Flume. We plan to extend the current query engine, and improve it to support complex interactive visualization queries. Another part is dedicated to storage facilities using a specific database system for engineering data. In Figure 4, some extensions are not defined for example: applications and complex queries. The Application component is dedicated to specific computations which run on the storage nodes; for example the computation of multi-resolutions objects. For the complex queries, not all of them have been yet selected, thus the future plugin has not been yet chosen. As stated in this Figure 4, the platform also supports different file systems: HDFS and Lustre for example. We also use the EDM database system, and provide a wrapper between the abstract file system layer in Hadoop and EDM.

IV. Conclusion

We introduce the VELaSSCo project. Simulations produce exponentially growing volumes of data, and it is not possible to store them anymore into existing IT systems. Therefore, VELaSSCo aims to develop new concepts for integrated end-user visual analysis with advanced management and post-processing algorithms for engineering applications, dedicated to scalable, real-time and petabyte level simulations. Data in this project are produced by two simulation sources: DEM and FEM applications. VELaSSCo is a solution to provide a complete platform to answer these needs.

We introduce the architecture of the platform, which is composed of a specific Hadoop distribution related to engineering data processing. The choice was made with respect to some requirements: support complex architectures, support multi-sources aggregation, query lead by visualization, scalability and extensibility. It is be composed of an open-source Hadoop distribution, using myHadoop and pre-installed extensions and scripts for visual queries. We plan to extend the storage by providing a plugin to use the EDM commercial database system as a file system. This software is an engineering database which supports large and complex engineering applications.

Our future work will be mainly focused on complex visual queries on Big Data, and more precisely on real-time streaming queries according to dynamic camera locations.

Acknowledgments

This work was supported in part by the EU FP7 project VELaSSCo, project number: 619439, FP7-ICT-2013-11.

References

[1] C. F. Claver, D. W. Sweeney, J. A. Tyson, B. Althouse, T. S. Axelrod, K. H. Cook, L. G. Daggert, J. C. Kantor, S. M. Kahn, V. L. Krabbendam et al., "Project status of the 8.4-m lsst," in Astronomical Telescopes and Instrumentation. International Society for Optics and Photonics, 2004, pp. 705–716.

[2] B. Lange and P. Fortin, "Parallel dual tree traversal on multi-core and many-core architectures for astrophysical n-body simulations," Euro-Par 2014, 2014.

[3] W. Dehnen, "A hierarchical o (¡ i¿ n¡/i¿) force calculation algorithm," Journal of Computational Physics, vol. 179, no. 1, pp. 27–42, 2002.

[4] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in ACM SIGOPS Operating Systems Review, vol. 37, no. 5. ACM, 2003, pp. 29–43.

[5] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008.

[6] Z. Fadika, E. Dede, M. Govindaraju, and L. Ramakrishnan, "Mariane: Mapreduce implementation adapted for hpc environments," in Grid Computing (GRID), 2011 12th IEEE/ACM International Conference on. IEEE, 2011, pp. 82–89.

[7] E. Dede, Z. Fadika, J. Hartog, M. Govindaraju, L. Ramakrishnan, D. Gunter, and R. Canon, "Marissa: Mapreduce implementation for streaming science applications," in E-Science (e-Science), 2012 IEEE 8th International Conference on, Oct 2012, pp. 1–8.

[8] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," ACM SIGOPS Operating Systems Review, vol. 41, no. 3, pp. 59–72, 2007.

[9] S. Krishnan, M. Tatineni, and C. Baru, "myhadoop-hadoop-on-demand on traditional hpc resources," San Diego Supercomputer Center Technical Report TR-2011-2, University of California, San Diego, 2011.

[10] A. Abouzeid, K. Bajda-Pawlikowski, D. Abadi, A. Silberschatz, and A. Rasin, "Hadoopdb: An architectural hybrid of mapreduce and dbms technologies for analytical workloads," Proc. VLDB Endow., vol. 2, no. 1, pp. 922–933, Aug. 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1687627.1687731

[11] P. Lindstrom, D. Koller, W. Ribarsky, L. F. Hodges, N. Faust, and G. A. Turner, "Real-time, continuous level of detail rendering of height fields," in Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. ACM, 1996, pp. 109–118.

[12] H. Hoppe, "Progressive meshes," in Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. ACM, 1996, pp. 99–108.